Finding Real Roots of Polynomials (Theory and Application in Computer Algebra)

AUTHOR(S):

平野, 照比古

CITATION:

平野, 照比古. Finding Real Roots of Polynomials (Theory and Application in Computer Algebra). 数理解析研究所講究録 2000, 1138: 234-239

ISSUE DATE:

2000-04

URL:

http://hdl.handle.net/2433/63811

数理解析研究所講究録
1138巻 2000年 234-239

Finding Real Roots of Polynomials

神奈川工科大学　平野　照比古(Teluhiko Hilano) *

1 Introduction

整数係数多項式の実数解を求める計算方法はいろいろな方法が知られている。たとえば [9] には多項式の複素数解を含めて解法を 29 の方法に分類し、そこの言葉によれば

> we present a comprehensive bibliography on roots of polynomials, covering (hopefully) most published work between the "Dawn of History" and 1994 .

の方針の元に多数の参考論文が掲げられている。

[6] では今後の数値計算技術はより高品質 (High Quality) でより信頼性のある (High Reliability) 方向が必要であると提案している。これに応えるためにこの小論では最近の数式処理システムを利用して (整数係数の) 多項式の与えられた区間における実根をすべて前もって与えられた精度以上で求めるプログラムを提供することを目的とする。また、要求された幅では根が分離できない場合には自動的にその精度を上げて結果を与える。

このような計算のアルゴリズムとして数式処理を用いた場合、Sturm の定理を用いる方法が有名である。Sturm の定理では与えられた多項式 $f(x)$ とその導関数 $f'(x)$ から、いわゆる Sturm 列を計算して、根の分離をする。この方法では Sturm 列の計算に非常に時間がかかる (らしい)。ここでは、このような深い定理を用いなくても十分に効率の良い方法を提示する。

2 アルゴリズム

アルゴリズムの本質は次の簡単な事実に基づいている。

定理 1 関数 $f(x)$ における導関数 $f'(x)$ の隣り合う二つの実根を α, β $(\alpha < \beta)$ とすれば $[\alpha, \beta]$ において $f(x)$ は単調である。したがって、この区間では $f(x) = 0$ の実根は高々一つしか存在しない。

*hilano@gen.kanagawa-it.ac.jp

与えられた関数が多項式の場合はこれを繰り返すことにより、最終的に定数関数になるのでここから逆に計算をたどることにより与えられた多項式の実根を求めることが可能となる。

3 インプリメントの方針

上の定理をそのまま応用すると $f(x)$ が重複因子を持つ場合、判定を正確にすることは出来ない。そこで与えられた多項式の無平方分解を利用する。無平方な多項式とその導関数は共通根を持たない。さらに、その実根の前後で関数は必ず符号を変えるので存在するかどうかの判定が確実に出来る。

実際の数式処理システムには次のような方針でインプリメントした。

- 数式処理では [4, p.428, 上から 5 行目] や [8, p.68,ptozp の解説を見よ] にもあるように有理数の計算が重いので、すべての計算を整数で行う。小数点以下 N 桁精度の計算をするためには与えられた多項式 $f(x)$ を $f\left(\frac{x}{10^N}\right)$ に置き換え、それの係数を整数化したものを用いる。
- $f_j(x)$ の実根はその存在範囲を他の実根と重複しないような区間で与える。

4 根の分離のアルゴリズム

根の分離に関するアルゴリズムの方針は次のとおりである。

$f(x)$ を与えられた関数とする。この関数の区間 $[a, b]$ に実根があるかどうかの判定をする。ただし、

- $f(x)$ と $f'(x)$ には共通根がない。
- b は上の区間における唯一の $f'(x) = 0$ 実根である。
- α と β は有理数で $\alpha \leq b \leq \beta$ を満たす。

が成立しているものとする。前節の方針により、α と β は整数として良い。

1. $f(a)$ の符号と $f(\alpha)$ の符号が異なれば区間 $[a, \alpha]$ に $f(x)$ の実根が存在する。
2. $f(a)$ の符号と $f(\beta)$ の符号が異なれば区間 $[a, \beta]$ に $f(x)$ の実根が存在する。
3. 区間 $[\alpha, \beta]$ の幅が 1 より大きいときは、この区間の幅を半分にして上記のことを行う。
4. 区間の幅が小さく出来ないときは ($\beta - \alpha = 1$ となっている。) 区間演算により $f(x)$ の区間 $[\alpha, \beta]$ における符号を定める。符号が定まればこの区間に $f(x)$ の実根はない。

5. $g(x)=f(\alpha+x)$ とおき、$g(x)$ の区間 $[0,1]$ における符号を区間演算で決定する。

6. 符号が定まらないときは精度を上げて繰り返す。

5 は重要な条件である。与えられた多項式によってはこの判定条件がないと不必要に途中の計算精度があがることがあった。たとえば竹島 [7] の場合、この判定条件を省くと計算時間が 10 倍以上になり、得られた解の精度を 30 桁指定しても、得られた解は 50 桁を超していた。このようなことが起こると計算の効率が落ちる。また、これらの条件をチェックする順番を変えただけでも計算時間に大きな変化が生じた。(一般には区間演算による計算の最終結果の区間の幅は、考えている区間の幅が狭いほうが狭い。また、同じ区間の幅を持っていても原点に近いほうが計算結果の幅が狭いことが経験上知られている。)

5 解の精度を上げるアルゴリズム

解の精度を上げるときの計算は整数の範囲における Newton 法を用いた。整数だけで計算を行うことと、近似解が単調減少か単調増大になるのかを前もってチェックしないので次のようなアルゴリズムを構成した。また、一度に目的の精度まで上げた多項式ではじめるのではなく Newton 法の二乗収束の性質を利用しして、精度を初期値から約 2 倍づつ上げていく方法をとった。

$f(x)=0$ の実根が区間 $[\alpha,\beta]$ にただ一つあり、それが単根であるとする。(ただし、α,β はともに整数とする。)

1. $\gamma=\alpha$ とおく。

2. $f'(\gamma)=0$ のときは Newton 法が適用できないので $\gamma^*=\dfrac{\alpha+\beta}{2}$(整数の範囲で計算する。) そうでないときは次のステップで γ^* を計算する。

3. $\gamma^*=\gamma-\dfrac{f(\gamma)}{f'(\gamma)}$ とおく。(ここの除法も整数の範囲で行う。)

4. $\gamma^*<\alpha$ または $\gamma^*>\beta$ のときは Newton 法が無効なので $\gamma^*=\dfrac{\alpha+\beta}{2}$ とする。

5. $\gamma^*=\alpha$ のときは Newton 法の計算が収束した可能性が大きいので $\gamma^*=\alpha+1$ とおく。

6. $\gamma^*=\beta$ の場合も同様の理由により $\gamma^*=\beta-1$ とおく。

7. $f(\gamma^*)$ の符号を調べ、$f(\alpha)$ の符号と同じならば $\alpha=\gamma^*$, 異なれば $\beta=\gamma^*$ とおく。

8. $\gamma=\gamma^*$ とおく。

9. $\beta - \alpha > 1$ であれば 2 に戻る。

このアルゴリズムでは区間の幅が必ず、減少するので有限回の後に必ず計算は停止する。Newton 法では解の存在する区間の片方の端から実際の解に単調に近づいてくるので [2, p.79 Theorem 4.8] 変化しなくなった場合の特別な扱いのところ 5,6 は重要である。

6 実行例と計算時間

この方針の元に Risa/Asir, pari-gp と pari ライブラリモード (in C) 上にインプリメントした関数は

```
findrealzero(Poly, Low, High, Prec)
findrealzeroall(Poly, Prec)
```

の二つである。 `Poly` は解くべき多項式、`Low` と `High` はそれぞれ根を求める区間の左端と右端の値、(`findrealzeroall` の場合は根の存在範囲を係数から自動的に評価するので指定は不要) `Prec` は根を求めるための小数点以下の桁数である。たとえば

```
findrealzero(x^2-2, -4, 4, 10)
```

とすれば区間 $[-4,4]$ にある $x^2-2=0$ の根を小数点以下 10 桁の精度で求める、つまり $\pm\sqrt{2}$ を小数点以下 10 桁まで求めることになる。この結果は

```
[[[-14142135624,10],[-14142135623,10],1],
  [[14142135623,10],[14142135624,10],1]]
```

となる。これは二つの区間

$$[-14142135624 \times 10^{-10}, -14142135623 \times 10^{-10}],$$
$$[14142135623 \times 10^{-10}, 14142135624 \times 10^{-10}]$$

にそれぞれ重複度が 1 の根が存在することを意味しているので

$$1.4142135623 \leq \sqrt{2} \leq 1.4142135624$$

となることが分かる。

例 2 次の表は $[0,1]$ 区間におけるルジャンドルの多項式 $P_n(x)\dfrac{1}{n!}\dfrac{d^n}{dx^n}(x^2-1)^n$ を $n = 5, 10, \ldots, 300$ と 5 次おきに根の有効精度を小数点以下 *30* 桁の精度で *pari-2.0.16beta* のライブラリーモード上で計算した場合の計算時間とそれぞれの計算終了後の *pari* のスタック消費量である。*(OS* は *FreeBSD 3.2*、 *CPU* は *Athlon 500MHz*、*256MB, pari* スタックサイズは *160MB)*

pari のライブラリーモードで高次の多項式を扱うことが出来るようにするためには、pari のスタック上に出来る途中のオブジェクトをいかに回収するか (ガーベッジコレクション) が問題であった。

Athlon 500MHz 256MB Memory FreeBSD 3.2 の場合

次数	CPU 時間 (sec)			pari スタック量		
	根の分離	精度向上	総合時間	根の分離	精度向上	合計
100	1.047	4.227	5.273	201536	294596	496132
200	12.023	32.914	44.938	1064408	1082076	2146484
300	56.547	138.297	194.844	2933260	2362644	5295904
400	181.844	327.812	509.656	6533140	4136992	10670132
500	428.891	643.156	1072.047	11999456	6403704	18403160
600	847.461	923.797	1771.258	19903764	9165084	29068848
700	1548.289	1777.453	3325.742	30470084	12418440	42888524
800	2675.000	2191.719	4866.719	44209772	16167148	60376920
900	4212.680	3787.289	7999.969	61972532	20407508	82380040

7 他のシステムとの比較

冒頭にも述べたようにここで問題としているような高次の多項式の実根をある程度の精度で確実に計算してくれる環境を提示するシステムは少ない。pari-2.0 にインプリメントされている `polroots` は数少ないインプリメントのひとつである。このアルゴリズムは [1] によれば Shönhage のアルゴリズムを Gourdon がインプリメントしたものである。この関数は複素根まですべて求める。pari の `polroots` は pari のスタックを 80MB の大きさにとった時 280 次ではスタックオーバーフローが起きて計算できなかった。これに比べ Risa/Asir ではそれほどのメモリーを必要としていないことが分かる。

Mathematica[5] における関数 `DSolve` で 20 次程度の多項式を解いてみたところすべての根が実数となるべきものがいくつかの根が複素数になってしまったので比較は行っていない。これらのシステムでは根の計算に浮動小数点を用いている。

謝辞

この研究にあたり、当初よりこの問題に関心を持っていただきました齋藤友克 (上智大学)、竹島卓 (富士通研究所)、近藤祐史 (詫間電波高専) の皆様にお礼を申し上げます。特に、竹島さんからは [7, p.18–19] で言及されている 103 次式の実根を Sturm の方法で求めた時間を教えていただき、この方法の改良する原動力となったことをここに申し添えま

す。この例はアルゴリズムの改良にも大変役に立ちました。また、いろいろな文献を教えていただいたことにも感謝いたします。

参 考 文 献

[1] Batut, C., Belabas, K., Bernardi, D., Cohen H., Olivier, M, *User's Guide to PARI-GP*, electrical document in pari-2.0.17beta.tar.gz at `ftp://megrez.math.u-bordeaux.fr/pub/pari/`

[2] Becker, T., Weispfenning, V., *Gröbner Bases, A Computational Approach to Commutative Algebra*, Graduate Texts in Math. 141, Springer, 1993

[3] Henrici, P., *Elements of Numerical Analysis*, John Wiley & Sons, 1963

[4] Knuth, D. E., *The Art of Computer Programming*, Vol. 2 *Seminumerical Algorithms*, 3rd Ed., Addison Welsley, 1998

[5] Wolfram, S., *The Mathematica book*, 3rd ed., Wolfram Media, Cambridge Univ. Press, 1996

[6] 伊理 正夫, 計算の品質, bit **28**(1996), 9,p.52–55

[7] 齋藤友克, 近藤祐史, 三好善彦, 竹島卓, Displaying Real Solution of Mathematical Equations, 数式処理, Vol.6, No. 2, pp.2-21 (1998)

[8] 齋藤友克、竹島卓、平野照比古: 日本で生まれた数式処理ソフト— リサ アジール ガイドブック、SEG 出版、1998 年

[9] J. M. McNamee *A bibliography on roots of polynomials*
`http://www.elsevier.co.jp/homepage/sac/cam/mcnamee/`